UNIVERSITY OF EXETER
Q-STEP CENTRE

Q-Step

A step-change in quantitative social science skills

Funded by the Nuffield Foundation, ESRC and HEFCE

Probabilistic logic and statistical inference

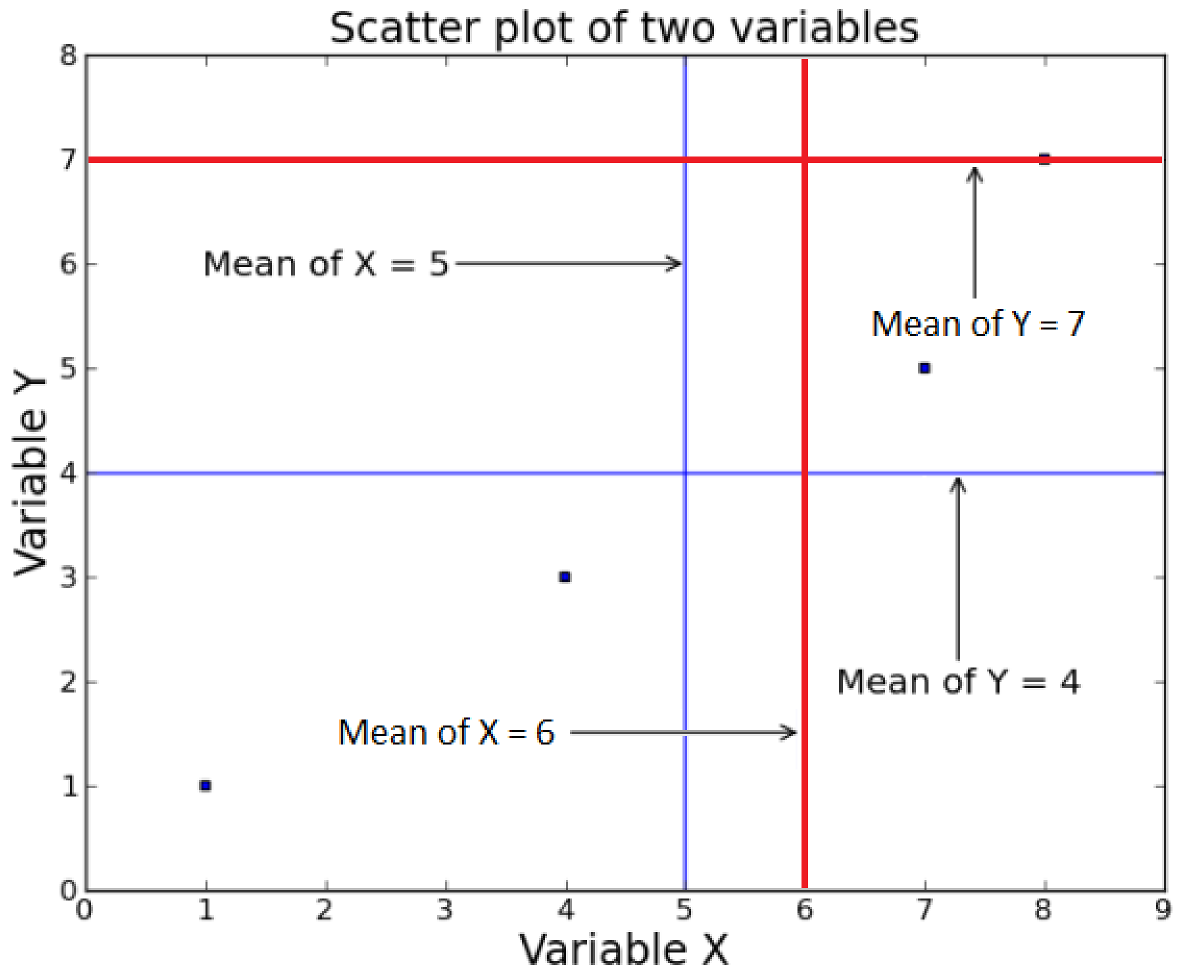
Part 1: Discrete variables

Lewys Brace
l.brace@Exeter.ac.uk

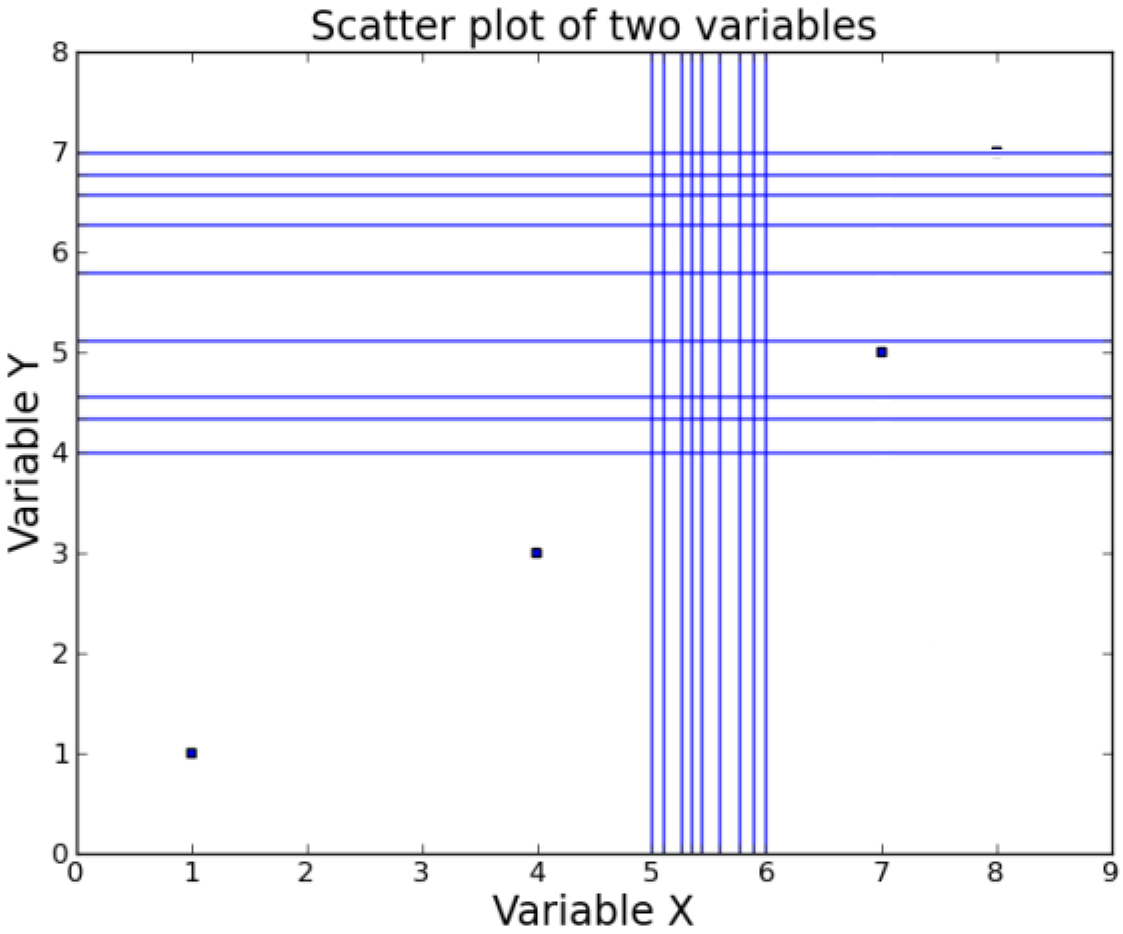
Statistical inference

- Last time we looked at correlation coefficients; how much of the variation in one variable can be explained by another?
- What if we wanted to actually predict the value one variable will take, based on our measurements of another variable?
- This is known as statistical inference.
- This is the process by which we go from measured data to probabilistic conclusions.

- Let’s go back to our example of fish-hours from the last lecture.
- We know that the mean of variable *X* (hours spent fishing) is 4 and the mean of variable *Y* (number of fish caught) is 5.
- However, the number of fish you catch per hour of fishing is likely to change from sample to sample.
- Furthermore, could you tell me the mean number of fish you would catch for all possible hours spent fishing?
- Indeed, if I went out fishing on another day, the chances are that I would get different means for the two variables.



- We could go ahead and keep measuring again, and again.
- We can see from the vertical lines that we would expect the mean number of fish caught to be between 4 and 7.



Probability

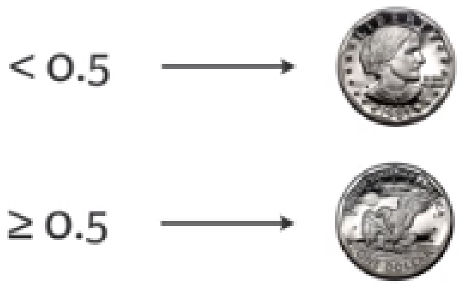
- We could not say for sure what the means for fish and hours would be for my next 50 days of fishing.
- But, if we were to draw a number of samples, then we could say that the means are likely to be closer to our 9 samples we have thus far than they are to be significantly greater, for example.
- In other words, given a set of data, you describe probabilistically what you would expect if that data was acquired over and over again.
- This is what Probabilistic reasoning is; it allows us to describe uncertainty.

Random number generators and ‘hacker statistics’

- In practice, we are going to think probabilistically with the aid of *hacker statistics*.
- The basic idea behind hacker statistics is that instead of literally re-sampling from the population, we use simulated repeated measurements in order to compute probabilities.

An example

- Let’s simulate the outcome of four coin flips.
- The aim here is to calculate the probability that we would get four heads out of four flips.
- We can make use of Numpy’s `random()` module in order to do this.
- We’ll use `np.random.random()` which draws a number between 0.0 and 1.0 in such a manner that all numbers in this range are equally likely to occur.



- These kind of ‘true/false’ experiments are known as *Bernoulli trials*.

Random number generator

- Random number generators, such as Numpy's `random()` module, works by starting with an integer, called a 'seed', and then generates random numbers in succession.
- The same integer will yield the same succession of numbers. Hence the term '*pseudo* random number generators'.
- If you want reproducible code, use `np.random.seed()`.

Simulating coin flips in Python

In:

```
import numpy as np
seed = np.random.seed(42)
random_numbers = np.random.random(size=4)
print "Random numbers:", random_numbers
heads = random_numbers < 0.5
print "Heads:", heads
sum_of_heads = np.sum(heads)
print "Sum of heads", sum_of_heads
```

Out:

```
Random numbers: [0.37454012 0.95071431 0.73199394 0.59865848]
Heads: [ True False False False]
Sum of heads 1
```

- However, we want to know the probability of gets heads if we were to run this experiment over and over again.
- Using a for loop, we this is easy to do:

In:

```
from decimal import *
import numpy as np
n_all_heads = 0
for i in range(10000):
    heads = np.random.random(size=4) < 0.5
    n_heads = np.sum(heads)
    if n_heads == 4:
        n_all_heads += 1
print Decimal(n_all_heads)/10000
```

Out:

```
0.0601
```

Hacker statistics probabilities

In short, hacker statistics involve:

- Determining how to simulate data.
- Simulating it many times.
- The probability is then approximately the fraction of trials with the outcome of interest.

Probability mass function (PMF)

- We simulated a story about someone flipping a coin.
- We did this to get the probability of possible outcome of the story.
- This set of probabilities is known as a *probability mass function*.
- A PMF is defined as the set of probabilities of discrete outcomes.

What is a distribution?

- The PMF is the property of a discrete probability distribution
- For now, think of a distribution as just a set of measurements, numbers, or data points.
- We could talk about the distribution of heads and tails results obtained when tossing a coin many times.
- Another distribution might be the final exam scores of every student in a particular school system.
- A third would be the predictions of global population in 2100 from 50 runs of a demographic simulation, each with a different random seed value.

Binomial distribution

- Our simulated coin flipping example from before corresponds to the *binomial distribution.*
- The ‘story’ behind such a binomial distribution would be:

“ The number r of successes in n Bernoulli trials with, probability p of success, is Binomially distributed.”

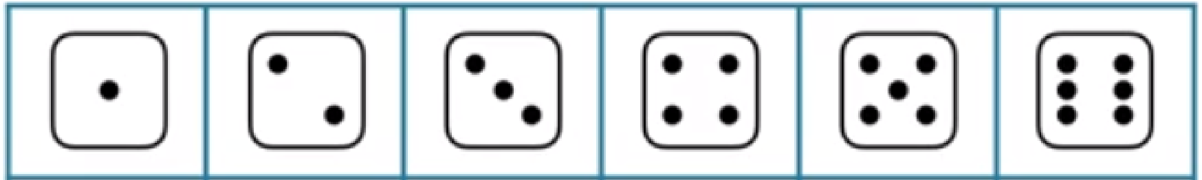
Our simulation matches this story:

“The number r of heads in 4 coin flips with, a probability 0.5 of heads, is Binomially distributed.”

Discrete uniform PMF

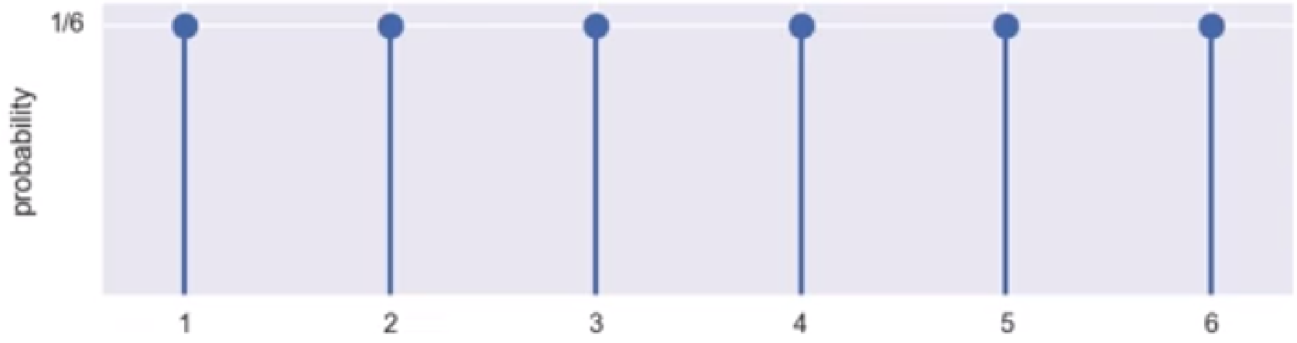
- If we considered a person rolling a single die once, for example.
- The values are discrete; i.e. you cannot role a 3.7.
- Each result has the same, or uniform, probability; 1/6.
- Thus, the PMF associated with this example is known as a *discrete uniform PMF*.

Tabular



1/6	1/6	1/6	1/6	1/6	1/6

Graphical



Sampling from the Binomial distribution

- We can sample from a Binomial distribution in Python with:

In:
```
from decimal import *
import numpy as np
flip = np.random.binomial(4, 0.5)
print flip
```

The number of Bernoulli trials (number of coin flips)

The probability of success (getting heads)

Out:
```
2
```

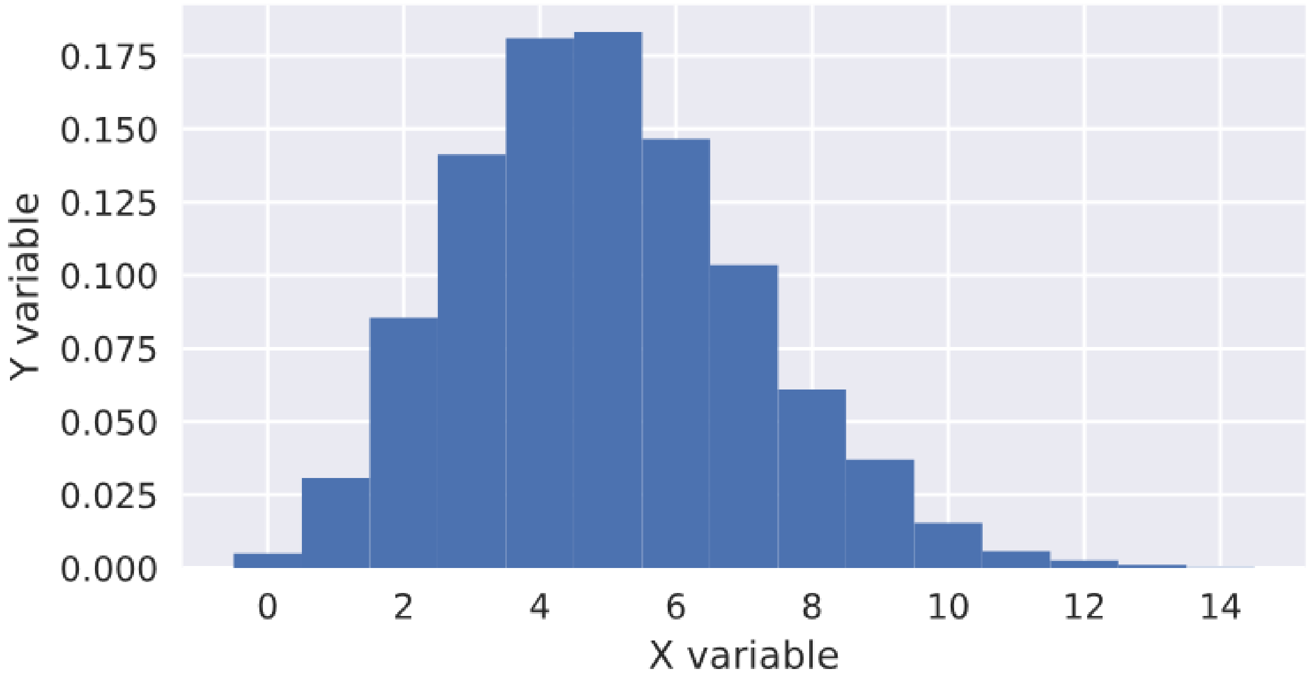
In:
```
from decimal import *
import numpy as np
flip = np.random.binomial(4, 0.5, size=10)
print flip
```

Out:
```
[1 2 4 4 2 3 2 1 3 2]
```

Plotting the binomial PMF

- Let's plot a PMF.
- To do this, we'll draw 10,000 samples from a binomial distribution, with 60 Bernoulli trials, and a probability of success of 0.1.
- Plotting the PMF is notoriously tricky, so it is better to plot it as a histogram or…

```
samples = np.random.binomial(60, 0.1, size=10000)
```



Binomial CDF

- The binomial CDF is just as informative as the PMF, and is easier to plot.

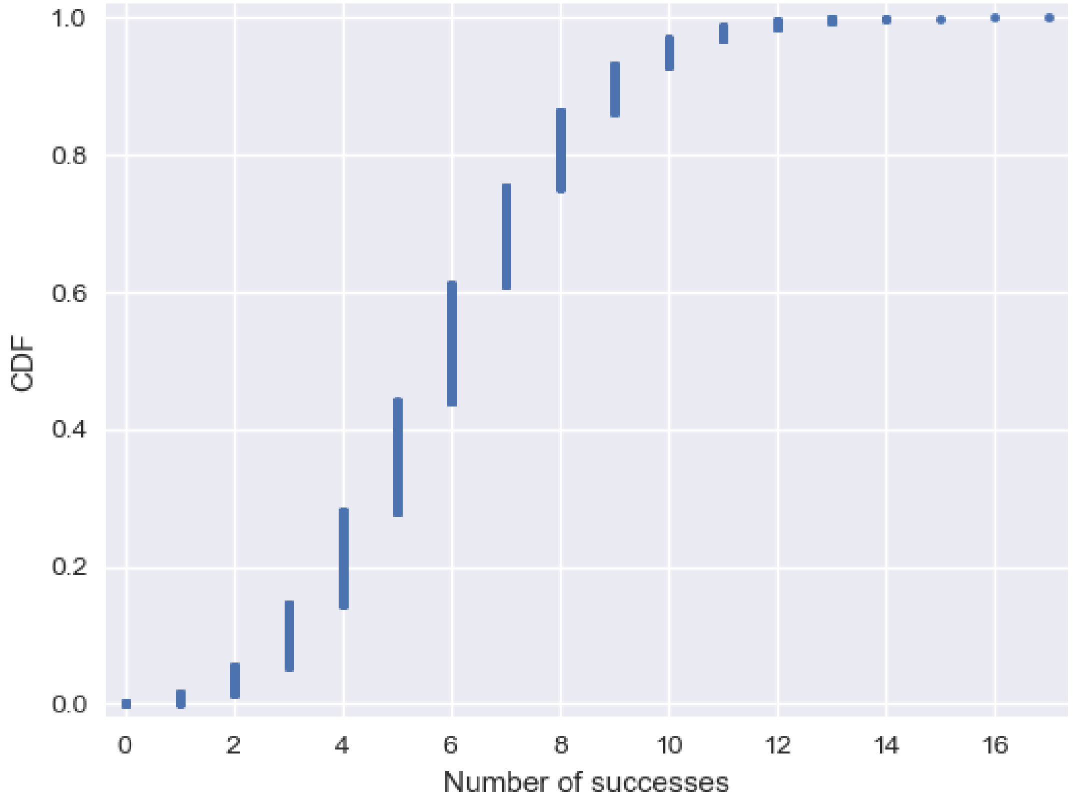
In:

```
from decimal import *
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

def ecdf(data):
    """Compute ECDF for a one-dimensional array of measurements."""
    # Number of data points: n
    n = len(data)
    # x-data for the ECDF: x
    x = np.sort(data)
    # y-data for the ECDF: y
    y = np.arange(1, Decimal(n)+1) / Decimal(n)
    return x, y

samples = np.random.binomial(60, 0.1, size=10000)
sns.set()
x, y = ecdf(samples)
_=plt.plot(x, y, marker='.', linestyle='none')
plt.margins(0.02)
_=plt.xlabel('Number of successes')
_=plt.ylabel('CDF')
plt.show()
```

Out:



Poisson processes and the Poisson distribution

- A Poisson process refers to the timing of the next event being completely independent of when the previous event happened.
- Many real-life process behave in this way; i.e.

  - Natural births in a given hospital

  - Meteor strikes

  - Hits on a webpage

  - Aviation accidents

Poisson distribution

- A Poisson distribution has one parameter:

  “The number, r, of arrivals of a Poisson process in a given time interval with average rate of λ arrivals per interval is Poisson distributed.”

- As an example, lets look at the average number of hits on a website in an hour:

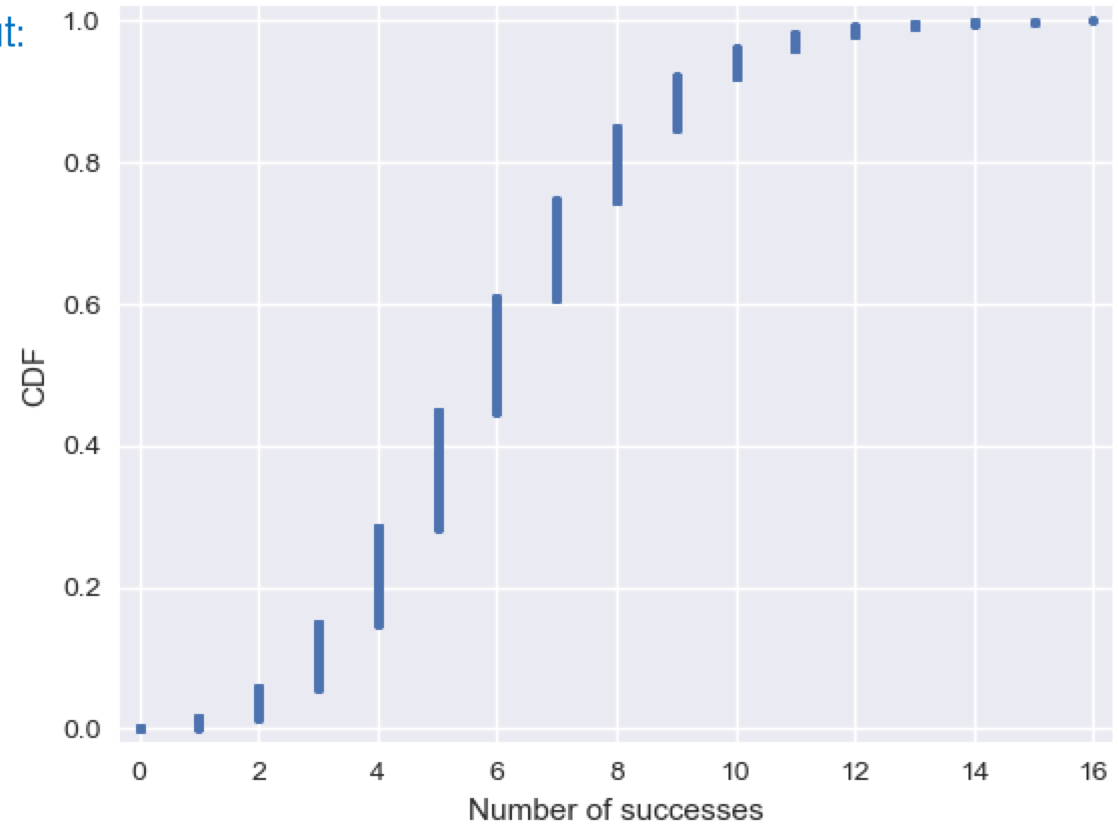
  “The number, r, of hits on a website in one hour with an average hit rate of 6 hits per hour is Poisson distributed.”

The Poisson CDF

In:

```
samples = np.random.poisson(6, size=10000)
x, y = ecdf(samples)
_=plt.plot(x, y, marker='.', linestyle='none')
plt.margins(0.02)
_=plt.xlabel('Number of successes')
_=plt.ylabel('CDF')
plt.show()
```

Out:



- For a given hour, we are most likely to get 6 hits (the average), but we may get as many as 10 or none.
- This graph looks like the binomial distribution, this is because….

Poisson distribution

- The Poisson distribution is a limit of the binomial distribution for low probability of success and large number of trials.
- In other words, for rare events.
- This makes sense if you think about the stories.
- Say we do a Bernoulli trial every minute for an hour, each with a success probability of 0.1.
- We would do 60 trials, and the number of successes is Binomially distributed, and we would expect to get about 6 successes.
- This is just like the Poisson story where we get six hits on a website.

- So, the Poisson distribution with arrival rate equal to *np* approximates a Binomial distribution for *n* Bernoulli trials with probability *p* of success (with *n* large and *p* small).
- Importantly, the Poisson distribution is often simpler to work with because it has only one parameter instead of two for the Binomial distribution.

Any questions?